UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2008 **Zdeněk Bejr**

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Přihlašovací systém pro oddíl orientačního běhu

Zdeněk Bejr

Bakalářská práce

2008

Zde bude zadání bakalářské práce 1/2

Zde bude zadání bakalářské práce 2/2

Poděkování:

Tímto bych chtěl poděkovat Ing. Zdeňku Šilarovi za cenné rady a připomínky, které mi poskytl v průběhu vypracování mé bakalářské práce.

SOUHRN

V této práci se zabývám vytvořením webové aplikace, sloužící k přihlašování orientačních běžců na závody. V první části popisuji použité technologie k vytvoření aplikace a jejich porovnání s ostatními technologiemi. Zaměřuji se především na bezpečnost, která je u takovéto aplikace velmi důležitá. Ve zbývající části řeším samotnou implementaci svého programu.

KLÍČOVÁ SLOVA

webové technologie, přihlašovací systém, PHP, MySQL, bezpečnost webu

TITLE

Registration Systém for an Orienteering Club

ABSTRACT

The work deals with creating a web application for registering orienteering runners for races. The first part describes the technologies used to create the application and their comparison with other technologies. An emphasis is put primarily on security, which is very important for this type of application. The remaining part is devoted to the program implementation itself.

KEYWORDS

web technology, registration system, PHP, MySQL, web security

OBSAH

1. Úvod ... 11
2. Výběr vhodného Webhostingu ... 12
2.1. Webový server ... 12
2.2. Výběr webhostingových služeb ... 12
3. HTML a HTTP ... 14
3.1. HTML ... 14
3.2. Protokol HTTP ... 15
4. Programovací jazyky ... 17
4.1. PHP ... 17
4.2. PERL ... 18
4.3. PYTHON ... 19
4.4. ASP.NET ... 19
4.5. Porovnání jazyků ... 19
5. Databáze ... 21
5.1. Úvod do databází ... 21
5.2. Příklady databází ... 23
5.2.1. MySQL ... 23
5.2.2. Postgre SQL ... 23
5.2.3. Fire Bird ... 23
5.2.4. Oracle ... 23
6. Použité technologie ... 24
6.1. MD5 (Message-Digest Algorithm) ... 24
6.2 SHA (Secure Hash Algorithm) ... 24
6.3 Regulární výrazy v PHP ... 25
6.4 Cookies ... 26
6.5 Relace ... 27
6.6 SQL injection ... 27
7. Analýza - normální formy ... 29
7.1 Nultá normální forma (0NF) ... 29
7.2 První normální forma (1. NF) ... 30
7.3 Druhá normální forma (2. NF) ... 30
7.4 Třetí normální forma (3. NF) ... 31
7.5 Čtvrtá normální forma (4. NF) ... 32
7.6 Pátá normální forma (5. NF) ... 32

8. Projekt přihlašovacího systému 33
8.1. Orientační běh 33
8.2. Důvod vzniku aplikace a požadavky 33
9. Vývoj aplikace přihlašovacího systému 35
9.1 Návrh databáze 35
9.2. Návrh a základní popis programu 36
9.3. Úvodní strana 36
9.3.1. Popis úvodní strany 36
9.3.2. Programové řešení hlavní strany 38
9.4. Uživatelské rozhraní 38
9.4.1. Popis uživatelského rozhraní 38
9.4.2. Implementace uživatelského rozhraní 40
9.5 Administrátorský účet 42
9.5.1. Popis administrátorského rozhraní 43
9.5.2. Implementace administrátorského účtu 44
9.6. Účet odesílatele 45
9.6.1 Popis rozhraní pro odesílatelský účet 45
9.6.2. Programové řešení odesílatelského účtu 47
10. Závěr 48

Seznam obrázků

Obr. 1: Graf dotazů na programovací jazyky ve vyhledávači Google.cz. [16] 20
Obr. 2: Princip zpracování příkazu jazyka SQL. [2] 22
Obr. 3: Ukázka aplikace – hlavní strana. 37
Obr. 4: Ukázka aplikace – přihlášené závody. 40
Obr. 5: Ukázka aplikace – administrátorský účet. 44
Obr. 6: Ukázka aplikace – odeslání závodu. 46

Seznam tabulek

Tab. 1: 0. Normální forma. 29
Tab. 2: 1. Normální forma. 30
Tab. 3: Špatně navržená tabulka 2. N.F.. 31
Tab. 4: Dekompozitovaná tabulka – reg. číslo. 31
Tab. 5: Dekompozitovaná tabulka – oddíl. 32

Seznam zkratek

ČSOB – Český svaz orientačního běhu

DTD – Document Type Definition

HTML – HyperText Markup Language

HTTP – Hypertext Transfer Protocol

MD5 – Message-Digest Algorithm

PHP – PHP Hypertext Preprocesor

SEQUEL – Structured English Query Language

SHA – Secure Hash Algorithm

SK LOB Nová Paka – Sportovní klub lyžování a orientačního běhu Nová Paka

SQL – Structured Query Language

WWW – World Wide Web

XML – eXtensible Markup Language

1. Úvod

V této bakalářské práci se budu zabývat vytvořením přihlašovacího systému pro orientační běh. Systém by měl sloužit členům SK LOB Nová Paka pro přihlašování na závody. Aplikace bude vytvářena tak, aby vyhovovala všem požadavkům a potřebám tohoto oddílu. Zároveň by měla vyhovovat normě ČSOB pro podávání přihlášek na jednotlivé závody.

Nejprve popíši různé webové technologie, které se používají při tvorbě podobných aplikací. Tyto technologie se pokusím krátce popsat a porovnat. Zejména se zaměřím na bezpečnost webových aplikací, která je u programů tohoto druhu klíčová.

V poslední části uvedu důvody vzniku přihlašovacího systému a popíši konečnou tvorbu mé aplikace. Budu se snažit uvést všechny důležité části zdrojového kódu.

Na závěr zhodnotím klady a zápory mnou vytvořené aplikace.

2. Výběr vhodného Webhostingu

Výběr vhodného webhostingu je jedna ze základních věcí při tvorbě každé internetové aplikace. Implementace se musí přizpůsobit tomuto webovému serveru. Nejedná se zde jenom o to, zda je nebo není na stránkách umístěna reklama, ale hlavně o použité technologie. Každý webový server poskytuje různé technologie a různou podporu. Proto jako první věc musíme najít vhodný webhosting.

2.1. Webový server

Webhostingové služby znamenají, když někdo nabízí místo a s tím spojené služby na serveru. Tento server je připojen k velmi rychlému internetu. Každý zákazník má na serveru vyhrazeno určité místo. Dnešní limity se nejčastěji pohybují u placených hostingů v jednotkách gigabajtů. Mezi další služby nabízené hostingem patří nejčastěji e-mail, ftp, různé statistiky atd.

Webový server přijímá požadavky od klienta (pomocí protokolu HTTP). Tyto požadavky vyřizuje a vrací odpověď, kterou je nejčastěji HTML stránka. Ta je buď předem připravená (statický obsah), nebo se vytváří podle požadavků klienta (dynamický obsah). Statické stránky jsou vráceny rychleji. Dynamické stránky ale mohou lépe reagovat na klientské požadavky. V praxi se oba způsoby kombinují – například pomocí cachování.

2.2. Výběr webhostingových služeb

Na začátku si náš oddíl stanovil, že chce vlastní doménu – „npa.cz“. Tuto doménu jsme chtěli umístit na placený webhosting. Důvody byly zřejmé, a to: spolehlivost serveru, rychlost, žádná reklama a už řečená vlastní doména. Kromě těchto základních požadavků jsem měl ještě jeden požadavek, a to spuštění zadaného skriptu v daném intervalu. To se ukázalo jako největší problém u těchto poskytovatelů a značně to zúžilo možný výběr. Mé požadavky byly:

- PHP (PHP ve verzi 5)

- MySQL (PHPMyAdmin)
- Mod_rewrite (optimalizace URL pro SEO)
- Minimum 2 GB místa
- E-mail (přesměrování emailů) – každý člen by měl v budoucnu mít jednotnou emailovou adresu ve tvaru jmeno.prijmeni@npa.cz
- Subdomény (ftp, web aliasy) – každý člen může mít vlastní web. Kromě toho má oddíl ještě svoji prezentaci (nezávislou na přihlašovacím systému). Proto je potřeba neomezený počet subdomén.
- CRON (nutnost spouštění skriptů v daném intervalu)

Po důkladném hledání na internetu, zejména na diskusních fórech, jsem se rozhodl pro webový server C4 (www.c4.cz). Nenašel jsem snad žádnou zápornou reakci ani připomínku. Mezi největší klady patří zákaznická podpora s velmi rychlou odezvou. V ceně hostingu je i doména, kterou sami zaregistrují. Podporuje vše výše zmíněného a navíc přidávají každý rok 500MB místa, což se při rozrůstající informační kapacitě může hodit zejména na fotografie.

3. HTML a HTTP

Protože moje bakalářská práce je webovou aplikací, chtěl bych zde vysvětlit základní technologie a princip fungování webových stránek. Popíši zde jazyk HTML, ve kterém jsou psány internetové stránky, a přenosový protokol http, který zajišťuje samotný přenos těchto stránek.

3.1. HTML

Je to značkovací jazyk pro hypertext. Zkratka pochází ze slova HyperText Markup Language. To je jazyk umožňující publikaci dokumentů na internetu.

V roce 1990 byl navržen Timem Bernerem-Lee spolu s protokolem HTTP – přenosový protokol hypertextu. Zároveň vznikl první webový prohlížeč, který byl pojmenován WorldWideWeb. V roce 1993 byl vytvořen první grafický webový prohlížeč Mosaic. Nyní se vyvíjí HTML, verze 5. Měla by být hotova v roce 2010. Finální verze, zbavená všech chyb, se odhaduje na rok 2022.

Jazyk funguje pomocí formátovacích značek. Mezi dvě značky se uzavře část textu, a tím se určí formátování tohoto textu. Část kódu obsahující počáteční značku, text a končící značkou koncovou se nazývá element. Součástí elementu mohou být další vnořené elementy. Například:

```
<b> toto je tučný text <i> a toto tučná kurzíva </i> </b>
```

Existují i nepárové tagy, jako je například <hr> (vodorovná čára) nebo
 (konec řádku). Tagy mohou obsahovat i atributy, které popisují jejich vlastnosti nebo nesou nějakou jinou informaci.

Struktura dokumentu psaného v jazyku HTML:

- Deklarace DTD – povinná od verze 4.01 , začíná direktivou <!DOCTYPE

- Kořenový element – reprezentuje celý dokument (<HTML> </HTML>)
- Hlavička dokumentu – obsahuje metadata vztažená k celému dokumentu. Ta mohou definovat název, kódování, klíčová slova, vývojový editor a další
- Tělo dokumentu – obsahuje vlastní text dokumentu (<BODY> </BODY>)

Příklad základní kostry HTML dokumentu:

```
<!DOCTYPE HTML PUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN">
<html>     //začátek kořenového elementu
<head>     //začátek hlavičky dokumentu
     <title> Název dokumentu </title>
</head>
<body>     //začátek těla dokumentu
     <h1> Nadpis stránky </h1>
     Tělo dokumentu …
</body>
</html>
```

3.2. Protokol HTTP

HTTP znamená Hyper Transfer Protokol. Je to internetový protokol určený pro komunikaci mezi HTTP serverem a terminálem (např. webový prohlížeč). Spolu s elektronickou poštou patří mezi nejvyužívanější protokoly na internetu. Obvykle používá port TCP/80.

Přenos je založen na principu dotazu a odpovědi. Nejprve se sdělí serveru požadavek a on odpoví souhrnem informací pomocí tzv. hlaviček (headers). Pokud byl v dotazu požadován nějaký soubor, který se na daném serveru nachází, je součástí odpovědí i obsah tohoto souboru. Když bude mít uživatel další dotaz na tento server, nelze zjistit, zda první dotaz souvisel s dotazem druhým. Proto se

tomuto protokolu říká bezestavový protokol. To znamená, že dotazy na sebe nemají žádnou návaznost.

Když dostane server požadavek přes protokol, může odpovědět třemi způsoby:

- Vrátit požadovaný dokument (kód 200)
- Oznámit, že se požadovaný dokument nachází jinde (kód začíná 3)
- Ohlásit jiný problém (kód začíná 4 nebo 5)

Každá odpověď od serveru je číslována trojmístným číslem, které přesně specifikuje nastalou událost. Za číslo se obvykle přidává vysvětlení, ale to není povinné. Některé webhostingy umožňují definování vlastních chybových stránek. Rozhodující je ale číslo kódu.

Zde uvedu nejčastější odpovědi serveru:

- **200** OK = server obsahuje požadovaný dokument a spolu s tímto dokumentem posílá kód 200.
- **301** moved permanently = znamená trvalé přesunutí dokumentu a klient přechází na novou adresu.
- **302** moved temporarily = přesunuto dočasně. U obou kódů začínajících 3 je na dalším řádku poslána hlavička Location, která udává nové URL.
- **404** not found = když klient zašle požadavek na obsah, který se na serveru nenachází.
- **403** forbidden = server zašle tento kód, když se snažíte o přístup na vám nepovolený obsah (nejčastěji nějaký adresář, který nemá právo výpisu).
- **500** internal server error = požadavky začínající 5kou signalizují problém na serveru. Chyba není na straně klienta.

4. Programovací jazyky

Jednou z nejdůležitějších věcí je výběr vhodného programovacího jazyka. Jazyky můžeme dělit na ty, co se vykonávají na straně serveru, a ty, co se vykonávají na klientském počítači. Skripty vykonávající se na straně klienta mají tu nevýhodu, že jejich zdrojový kód si může kdokoliv přečíst. Každý, kdo si stahuje HTML kód, si zároveň stahuje zdrojový text programu. Mezi největší výhody druhé skupiny patří, že je skript spuštěn přímo na serveru a uživateli se odesílá již zpracovaná HTML stránka. Odpadají tím případné problémy s nekompatibilitami daného vývojářského nástroje. Stačí jednoduchý prohlížeč HTML. Další veliká výhoda je, že se nikdy neodesílá zdrojový kód stránky. Mezi nevýhody této kategorie patří, že tyto programy nemohou být interaktivní na akce uživatelů. Server posílá statickou stránku, a proto zde nemůžeme zachytit např. událost kliknutí myši.

Skriptovací jazyky nebo též interpretované jazyky pro svůj běh potřebují tzv. programový interpret. Bez tohoto interpretu nelze skripty spouštět. V tom se zásadně liší od Pascalu, C/C++, které lze po kompilaci samostatně spouštět.

4.1. PHP

Je to skriptovací programovací jazyk, primárně určený pro vytváření dynamických internetových stránek. Autorem tohoto jazyka je Rasmus Lerdorf, který jej vytvořil v roce 1994 pro svoji vlastní potřebu počítání přístupu k jeho stránkám. Vytvořil ho přepsáním skriptu Perlu do jazyka C. Dal mu jméno Personál Home Page. V roce 1997 přepsali parser Z. Suraski a A. Gutmans a vytvořili základ pro PHP3. Dochází k největšímu rozmachu jazyka. Od této verze byl také změněn název na současnou podobu PHP Hypertext Preprocessor. V roce 2000 vychází verze 4 a o další čtyři roky později verze PHP 5. Ta již podporuje objektově orientované programování, které usnadňuje správu rozsáhlých projektů, neboť objektově orientovaný zdrojový kód je daleko přehlednější a dá se rovněž lépe spravovat. Nová verze přinesla i lepší podporu XML souborů a lepší zpracování ohledně zvýšení zabezpečení zdrojového kódu vůči chybám.

V PHP jsou naprogramovány i větší projekty, jako např. webový server Centrum.cz nebo třeba Wikipedie.

PHP je nyní dostupné pro platformy Linux, Windows i Macintosh. Všechny verze PHP jsou zcela zdarma. Obsahují rozsáhlé knihovní funkce podporující zpracování textu, grafiky, práci se soubory, přístup k databázím a mnohé další. Když se použije PHP v kombinaci s databázovým serverem a webovým serverem, vzniká velmi mocný nástroj pro tvoření internetových stránek. Vžila se zkratka LAMP – spojení Linux, Apache, MySQL a PHP nebo Perl.

PHP je vepsáno do HTML kódu a interpreter na místo těchto kódů vkládá výsledek překladu. K uživateli se tak dostane jenom čistá stránka HTML.

```
<body>
  Toto je normální text.
  <?php
     echo “Toto je text tištěný jazykem PHP”;
  ?>
</body>
```

4.2. PERL

Zkratka znamená Practical Extraction and Report Language. Stal se předním skriptovacím jazykem, ale není určen jenom pro webové vývojáře. Perl je interpretovaný jazyk vytvořený Larry Walsem v roce 1987, původně nástroj na zpracování textů. Po uvolnění interpretu se rychle rozšířil a dnes je z něho skutečný programovací jazyk umožňující objektově orientované programování, programování ve více souběžných vláknech, modulární programování atd. Původně byl vyvinut pro UNIX, ale v současné době je jedním z nejpřenositelnějších jazyků. Podporuje Windows, MS-DOS, Macintosh atd.

Ve srovnání s jazykem PHP je tento jazyk trochu složitější a vytváří delší skripty. Hodí se spíše na větší projekty nežli PHP.

4.3. PYTHON

Python je interpretovaný objektově orientovaný programovací jazyk. V roce 1990 jej vyvinul Guido van Rossum. Tento jazyk je zařazován do skriptovacích jazyků, ale umožňuje tvorbu velkých aplikací včetně grafického rozhraní. Podporuje procedurální, objektové i funkcionální programování. Díky tomu vzniká krátký, dobře čitelný kód. Python se snadno vkládá do jiných aplikací, kde pak slouží jako jejich skriptovací jazyk.

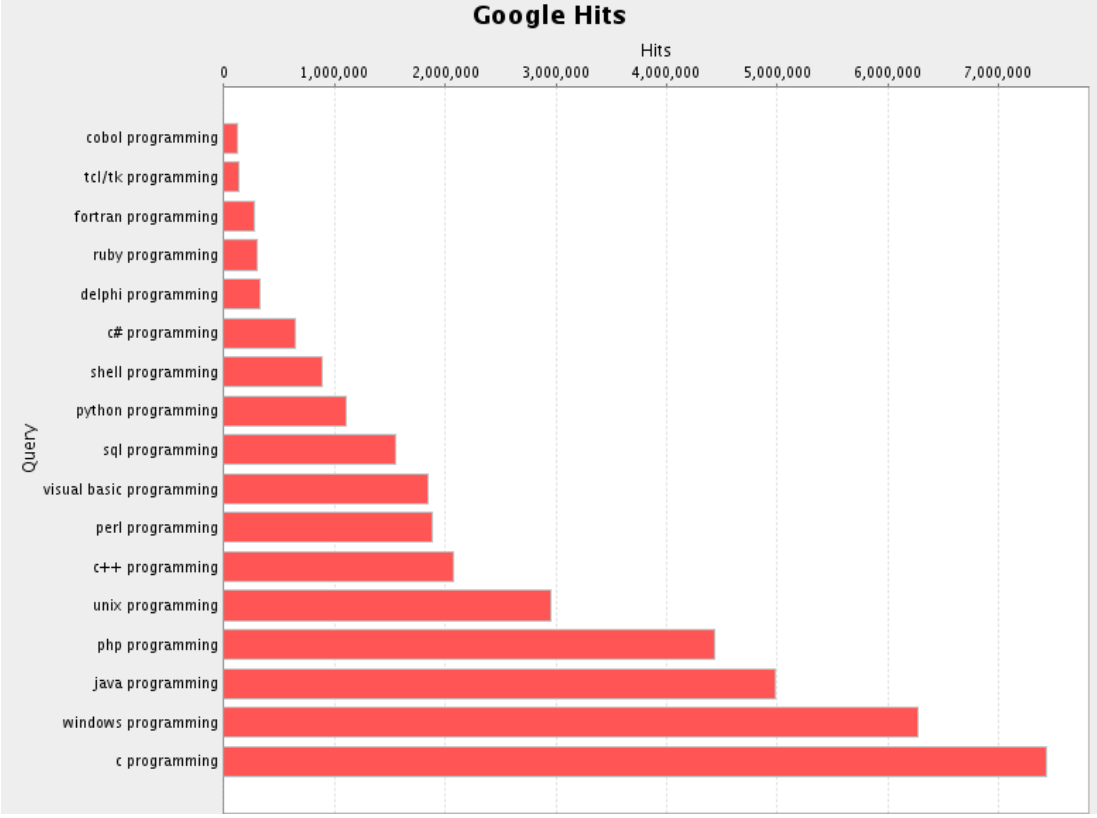
Výkon tohoto jazyku je na velmi dobré úrovni. Kritické knihovny byly implementovány do jazyka C a díky tomu jsou 3 až 5 krát rychlejší než PHP. Tento jazyk byl napsán v jazyce C a v tomto jazyku probíhá i další vývoj Pythonu. Je vyvíjen jako Open Source projekt, který zdarma nabízí instalační balíky pro většinu platforem.

4.4. ASP.NET

Byl vytvořen firmou Microsoft jako nástupce starého ASP (Active Server Pages), ale kromě názvu nemají mnoho společného. ASP se dnes již skoro nepoužívá. ASP.NET je soubor webových technologií, které patří do .NET frameworku. Aplikace založené na ASP.NET jsou rychlejší, protože bývají předkompilovány do souborů .dll. Jiné, čistě skriptovací jazyky parsují stránku při jejím každém zavolaní.

4.5. Porovnání jazyků

Jen velmi těžko se dá hodnotit, který jazyk je lepší a který horší. Každý má své výhody a nevýhody. Jazyky jsou různě implementovány a mají různou syntaxi, proto se každý hodí víc na určitou oblast vývoje. Určitým měřítkem oblíbenosti a počtu využívání by mohl být graf zadávání dotazů na jednotlivé jazyky do celosvětově nejpoužívanějšího vyhledávače Google.cz.



Obr. 1: Graf dotazů na programovací jazyky ve vyhledávači Google.cz. [16]

5. Databáze

Stejně důležitou částí, jako je výběr programovacího jazyka, je i výběr použité databáze. V této kapitole bych chtěl vysvětlit základní princip a uvést několik nejpoužívanějších databází.

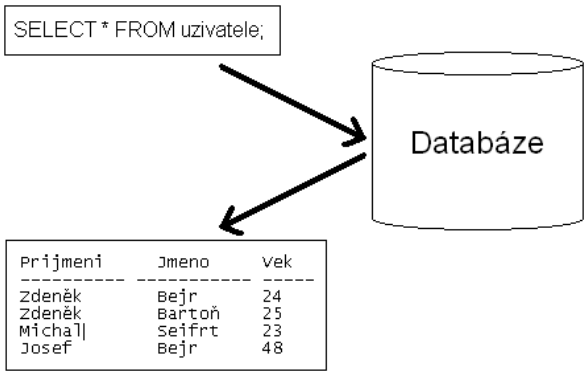
5.1. Úvod do databází

Databáze je místo, kam ukládáme data. Relační databáze se skládají z několika tabulek, které jsou mezi sebou různě provázány. Sloupce v tabulce musí být jedinečné názvem a každý obsahuje svůj datový typ podle dat, která chceme ukládat. Každá tabulka se skládá z jednotlivých řádků, které označujeme jako záznam.

Oproti souborům mají několik podstatných výhod:

- Databáze poskytuje rychlejší přístup k datům než soubor.
- Databáze umožňuje přímý přístup k datům.
- Databáze má zabudovaný mechanizmus pro paralelní přístup k datům.
- Databáze umožňuje dle dotazů vracet množiny dat, která přesně vyhovují zadaným kritériím.
- Databáze má zabudovaný systém uživatelských práv.

Pro většinu dnešních databází se používá jazyk SQL (Structured Query Language). Byl vyvinut firmou IBM, tehdy byl označován jako SEQUEL (Structured English Query Language). Cílem bylo vyvinout jazyk blízký angličtině pro práci s daty v databázích. Později na vývoji začalo spolupracovat více firem. SQL je dotazovací jazyk a základní princip komunikace si můžeme prohlédnout na obrázku.



Obr. 2: Princip zpracování příkazu jazyka SQL. [2]

Klient zformuluje a položí svůj dotaz a databázový server na něj odpoví (výstupní množina dat). Jazyk SQL připomíná klasický přirozený (anglický) jazyk, má ale přesně definovaná syntaktická a lexikální pravidla. Z pohledu serveru se příkaz SQL přenese, dekóduje, optimalizuje a provede.

Příkazy jazyka SQL můžeme rozdělit do čtyřech skupin:

- **Data Definition Language** (DDL, jazyk pro definici dat) - pomocí příkazů této skupiny můžeme definovat, měnit a rušit různé databázové struktury, jako jsou například tabulky, indexy, triggery apod.
- **Data Manipulation Language** (DML, jazyk pro manipulaci s daty) - do této skupiny patří příkazy pro manipulaci s daty. Jsou to příkazy pro vkládání, aktualizaci, mazání dat a samozřejmě také dotazovací příkaz SELECT.
- **Data Control Language** (DCL, jazyk pro správu dat) - tato skupina zahrnuje speciální příkazy pro řízení provozu a údržby databáze.
- **Transaction Kontrol Commands** (TTC, příkazy pro řízení transakcí) - vedle nejznámějších skupin příkazů jazyka SQL můžeme vyčlenit také skupinu příkazů pro řízení transakcí.

5.2. Příklady databází

5.2.1. MySQL

Byla vyvinuta švédskou firmou MySQL AB. Patří mezi multiplatformní databáze. Komunikuje pomocí jazyka SQL, přičemž tento jazyk trochu rozšiřuje. Je k dispozici pod dvojím licencováním - jako bezplatná pod licencí GPL i jako placená verze. Databáze neobsahuje takové množství funkcí a možností jako její konkurenti, je ale stavěna jako velmi rychlá databáze. Do verze 5 neobsahovala pohledy, triggery, úložné procedury atd. Dnes je pravděpodobně nejrozšířenější databází vůbec.

5.2.2. Postgre SQL

Tento server vyniká stabilitou, rychlostí a dobrou podporou pokročilých technologií. Některé webhostingy ho mají v nabídce vedle MySQL. PostgreSQL je šířena pod BSD licencí. Tato licence umožňuje neomezené bezplatné používání, modifikaci a distribuci. Má více možností než MySQL, je ale o něco pomalejší. Umožňuje běh uložených procedur napsaných v několika programovacích jazycích jako jsou: C, Python, Perl a další. Předností tohoto systému je rozšiřitelnost. PostgreSQL může být bezproblémově rozšiřován o nové datové typy, funkce, operátory, procedurální jazyky atd.

5.2.3. Fire Bird

Je multiplatformní databáze pracující na operačních systémech Linux, Windows a dalších. Mezi její výhody patří široká podpora ze strany vývojových nástrojů, především Delphi, .NET nebo Java. Má velkou propustnost při víceuživatelském zpracování a ohromnou podporu pro úložné procedury a spouště. Za vývojem této databáze stojí firma Borland.

5.2.4. Oracle

Na rozdíl od výše zmíněných databází je Oracle komerční projekt, pravděpodobně leader ve světě databází. Jejich řešení se stalo jakýmsi nepsaným pravidlem pro velké systémy. Je to moderní multiplatformní databázový systém.

6. Použité technologie

Tato kapitola se zabývá teoretickým popisem některých funkcí, které používám v programu. Jsou to části, které si zaslouží bližší a podrobnější vysvětlení. Většina se týká důležité bezpečnosti webu.

6.1. MD5 (Message-Digest Algorithm)

Je to funkce, která vytváří ze vstupního (libovolně velkého) množství dat výstupní množství dat fixní délky (otisk). Tento otisk má velikost 128 bitů. Někdy je otisk označován jako hash nebo miniatura. Jeho zásadní vlastností je, že malá změna na vstupu provede velkou změnu na výstupu (zcela odlišné otisky). Algoritmus MD5 je popsán standardem RFC1321.

Byl vytvořen Ronaldem Rivestem v roce 1991 a nahradil stávající MD4. V roce 1996 se objevila vada v návrhu a i přes to, že nebyla zásadní, kryptologové doporučili používat jiné algoritmy. V roce 2004 jej kryptologové z Japonska (později z ČR) označili jako nebezpečný, neboť objevili zpětný algoritmus, kterým je možno vysledovat přibližnou množinu vstupních znaků. Příklad výstupu algoritmu MD5 při malé změně vstupního textu.:

```
MD5(“Studuji obor IT“) = 962e8537caa3254ff0efc435ab7fc2c7
MD5(“Studuji obor it“) = 397e18ecdafbcf8ae1fad2415472536e
```

6.2 SHA (Secure Hash Algorithm)

SHA navrhla organizace NSA (Národní bezpečnostní agentura v USA). SHA1 vytváří otisk o velikosti 160 bitů. Algoritmus SHA1 je novější a ne tak chybový jako MD5. Je považován za jeho nástupce. Byl ale rovněž prolomen – ovšem pouze hrubou silou (metoda Brute force). Zatím nebyl žádný zpětný algoritmus vyvinut. V PHP se používá podobně jako MD5. Příklad otisku SHA1:

```
SHA1(“Studuji obor IT“) =
401e2f477c3c1f8b8cc78f1670ac103148c3732f
```

6.3 Regulární výrazy v PHP

Představují velmi povedený, ale bohužel ne právě jednoduchý způsob práce s řetězci. Představme si, že hledáme v textu rodné číslo. Pokud použijeme příkaz regexp(), je nalezení řetězce celkem snadné. Do podmínky pro vyhledání se zadá: `[0-9]{6}/[0-9]{4}`. Bez použití regulárních výrazů by bylo nalezení vhodného řetězce poměrně zdlouhavou záležitostí.

První pokusy o implementaci regulárních výrazů se objevily v Unixu. Používali je textový editor Ed a známý program Grep. Dnes se dají regulární výrazy pro vyhledání a nahrazení řetězců použít ve všech novějších textových editorech.

Velké úsilí o prosazení regulárních výrazů bylo vyvinuto u programovacích jazyků, zejména u jazyka Perl. Tam se rozšířily tak, že se vytvořil (upravil) standard POSIX pro regulární výrazy. Vznikla nová knihovna s názvem PCRE (Perl Compatible Regular Expressions), která se dnes používá u všech větších aplikací, jako je například Apache, Python, KDE nebo PHP.

PHP podporuje dvě různé knihovny regexp. První je standardu Posix 1003,2 a knihovnu s funkcemi kompatibilními s jazykem Perl (preg_*). První knihovna je standardní součástí jazyka PHP. Co se týče funkce preg_*, je nutné doinstalovat knihovnu PCRE.

Uvedu zde ještě příklad ověření platné emailové adresy v PHP, pro lepší názornost použití regulárních výrazů:

```
$email= “ydenek@post.cz”;
$reg_vyraz= ‘/[A-Z0-9.%_-]+@[A-Z0-9._%-]+/i’;
If(preg_match($email_match, $email_str)){
echo ”Platná emailová adresa !”; }
```

Regulární výrazy využívám ve své práci k ošetření všech vstupů, kde potřebuji získat přesný formát dat. Jedná se zejména o emailové adresy a formát data (datum závodu, datum přihlášky na závod).

6.4 Cookies

Představují způsob, pomocí něhož má webový server možnost ukládat na počítači klienta informace. Tou mohou být například nastavení, která zde uživatel provedl (při poslední relaci) a která mají být uložena pro další návštěvu. V souboru cookie se kromě vlastního obsahu nachází také informace o době platnosti a odesílateli. Maximální velikost souboru cookie se pohybuje okolo 4KB. Na klientovi se ukládají cookie v profilu každého uživatele zvlášť. Tím se zajistí, aby měl přístup ke cookie pouze přihlášený uživatel. Problémy mohou nastat všude tam, kde se sdílí uživatelský profil (internetové kavárny, atd.).

Cookies se hodí zejména k déletrvajícím operacím, jako je například přesun údajů přes několik stránek (nakupování v obchodě).

Existuje několik způsobů, jak data dočasně uložit:

- Údaje se vloží do skrytých zadávacích políček a přenesou se ze stránky na stránku. Tento způsob je ale poměrně složitý a může být zdrojem chyb.
- Údaje se přenáší pomocí cookies. Internetový prohlížeč musí mít v zabezpečení cookie povoleno.
- Pomocí relací. U klienta se nic neukládá, vše je uloženo na serveru, takže nezáleží na nastavení v prohlížeči. Je to asi nejlepší způsob z výše uvedených.

Velikou nevýhodou cookies je, že se na ně nedá spolehnout. Nelze totiž zaručit, že internetový prohlížeč po ukončení relace cookies smaže nebo zda během relace nesmaže uživatel cookies sám. Nedoporučuje se spoléhat na cookies u citlivých operací.

6.5 Relace

Relace pracují odlišně od cookies. Pro každou relaci (Session) se klientovi přiřadí nějaký jednoznačný identifikátor (Session ID). Pomocí tohoto identifikátoru server pozná jednotlivé relace. Všechny informace zůstávají na serveru. To je výhodné jak z hlediska zabezpečení, tak z hlediska práce s proměnnými.

I u relací je problém, jak server naváže spojení mezi session ID a klientem. PHP standardně ukládá ID do souboru cookies s názvem PHPSESSID (název se dá nastavit v souboru php.ini). Je to nejlepší způsob. Programátor se nemusí starat o správné předávání session ID, protože to je uloženo v souboru cookie. Výhoda proti cookies je, že všechny informace zůstávají na serveru. Když má uživatel soubory cookie zakázány, předá se session ID jako parametr. Adresa URL se rozroste právě o toto ID. To není zcela bezpečný způsob a v nových verzích PHP je tato možnost vypnuta (konfigurační soubor PHP). Na veřejně přístupných PC se dá session ID vyhledat pomocí historie stránek a když si uživatel přidá stránku do oblíbených, pak na ni bude přistupovat se stále stejným ID. Tento způsob by se měl používat jenom v bezpečném prostředí, jako je třeba intranet. Abychom mohli relace používat naprosto běžným způsobem, museli bychom přepracovat protokol HTTP. Tento protokol je totiž bezestavový. Po vyslání požadavku od klienta webový server tento požadavek zpracuje a odešle zpět výsledek. Pokud klient pošle další dotaz, server nevidí žádnou spojitost mezi prvním a dalším dotazem.

Slušným základem je, aby se sledovala IP adresa a verze prohlížeče. Ale i tyto proměnné se dají změnit, proto se nejedná o konečné řešení.

6.6 SQL Injection

Pod tímto pojmem si můžeme představit podvržení vstupních dat tak, aby byl nějakým způsobem pozměněn výsledek SQL dotazu. Nejčastější případy útoků probíhají přes neošetřené formuláře na webových stránkách. V dnešní době je na internetu spousta webů, které nejsou ošetřeny proti podvržení škodlivých dat. Ukázka SQL injection:

```
mysql_query= „SELECT jmeno, prijmeni FROM users WHERE email =
‘.$zadany_email’”;
```

Příkaz najde jméno a příjmení podle emailu. Když však někdo zadá:

```
neco‘; DROP TABLE users; --
```

tento příkaz smaže tabulku users. Když se chceme takovýmto „podstrčeným“ příkazům vyhnout, musíme escapovat nebezpečné znaky. Na straně databáze pak můžeme omezit útok vhodným nastavením práv uživatele (nesmí používat některé příkazy atd.). Proti útokům rovněž pomáhá vypnutí chybových hlášek. Pokud by útočník narazil na nějaký neošetřený kód, nedozvěděl by se o tom.

Pro escapování nebezpečných znaků v PHP máme tyto funkce: stripshlases, addslashes. Ty změní speciální znaky (“, ‘, \ a nulový bajt) tak, aby ztratily svůj význam. Pomocí funkce addslashes se před každý takovýto znak vloží znak zpětného lomítka, takže jej PHP bere jako normální znak. Problém může nastat, když zavoláme tuto funkci na řetězec, který tyto znaky už obsahuje. To se může stát celkem lehko. Tvůrci PHP vložili funkce (Magic Quotes), kde pro všechny vstupy, pocházející z externích zdrojů (metody GET a POST), se použije automaticky zpětné lomítko. To ovšem znamená, že se nastavení Magic Quotes musí testovat. Příklad otestování stavu Magic Quotes v PHP.

```
if(get_magic_quotes_gpc())
```

Vrací true, když je funkce zapnuta. V tom případě bychom již neměli vstup ošetřovat funkcí addslashes. Funkce Stripshlases dělá přesný opak. Ta odstraňuje z řetězce všechny znaky zpětného lomítka.

7. Analýza - normální formy

Před začátkem vývoje aplikace, která využívá databázi, musíme nejprve navrhnout samotnou strukturu této databáze. To, jak se databáze navrhne, má vliv na její celkovou výkonnost, jak jednoduše nebo složitě se bude spravovat a hlavně jak flexibilní bude (přidávání / ubírání položek). Chyby, které vytvoříme při návrhu, se napravují jenom velmi složitě.

Tabulky by neměly obsahovat žádná redundantní data. Pokud se vyplňují stále ty samé údaje, je něco špatně. V tabulkách by se ani neměly vyskytovat sloupce typu zbozi_1, zbozi_2, zbozi_3, … Může se stát, že zákazník si koupí jenom jedno zboží (tabulka je zbytečně složitá a pomalá) a nebo v horším případě si bude chtít objednat třeba šest druhů zboží. V tom případě narazíme na těžko řešitelný problém. Proces, který odstraňuje výše zmíněné nedostatky, se nazývá normalizace. Ta odstraňuje redundantní data a zefektivňuje práci. Mělo by platit, že v čím vyšší normální formě se tabulka nachází, tím lépe by se s ní mělo pracovat. Toto pravidlo neplatí zcela vždy a někdy se od normalizace vyšších forem upouští z důvodu rychlosti práce s tabulkou.

7.1 Nultá normální forma (0NF)

Tabulka je v nulté normální formě, pokud alespoň jedno pole obsahuje více než jednu hodnotu, tedy není atomické.

Jméno	Adresa
Bejr Zdeněk	Studentská 1327, 509 05 Nová Paka
Mokrý Lubomír	Gebauerova 15, 056 15 Praha
Bartoň Zdeněk	Sejkorova 148, 065 17 Brno

Tab. 1: 0. Normální forma.

Tabulka neumožňuje například třídění podle jména nebo města. Pomocí jazyka PL/SQL by to určitě šlo, ale bylo by to velmi pracné.

7.2 První normální forma (1. NF)

Tabulka splňuje první normální formy tehdy, pokud:

- Jsou odstraněny všechny sloupce se stejným obsahem.
- Pro každou skupinu navzájem spojených dat se musí vytvořit nová tabulka.
- Každý záznam se identifikuje pomocí tzv. primárního klíče.

Příklad tabulky z 0NF převedenou na 1NF.

ID	Jméno	Příjmení	Ulice	Město	PSČ
1	Zdeněk	Bejr	Studentská 1327	Nová Paka	509 01
2	Lubomír	Mokrý	Gebauerova 15	Praha	506 15
3	Zdeněk	Bartoň	Sejkorova 148	Brno	501 02

Tab. 2: 1. Normální forma.

Zde jsou všechny atributy z praktického hlediska již dále nedělitelné. Teoreticky by se dalo ještě rozdělit číslo popisné od názvu ulice, ale to se nedělá, protože se většinou vypisují společně. Třetí pravidlo říká, že každý záznam musí být identifikovatelný pomocí klíče. Klíč by měl být co možná nejkratší a nejjednodušší. V našem případě je to sloupec ID. Ne vždy se ale musí dělat nový sloupec s pořadím položek, stačí nalézt sloupec, který se nemůže opakovat, například rodné číslo.

7.3 Druhá normální forma (2. NF)

Tabulka splňuje druhou normální formu tehdy, když:

- Splňuje první normální formu.
- Každý sloupec (vyjma primárního klíče) musí být zcela závislý na celém primárním klíči.

Druhá normální forma se proto týká jen těch tabulek, které mají více primárních klíčů. Příklad špatné tabulky pro druhou normální formu.

Registrační číslo	Oddíl	Licence	Zkratka oddílu
8401	Nová Paka	C	NPA
8151	Hostinné	D	HOS
9001	Nová Paka	A	NPA

Tab. 3: Špatně navržená tabulka 2. N.F.

Tabulka má primární klíč tvořený dvěma sloupci Registrační číslo a Oddíl. Navíc obsahuje i data zadávaná opakovaně, což je typickým znakem tabulek nesplňujících druhou normální formu. Aby tabulka splňovala 2NF, musí se provést dekompozice na dvě tabulky. Každá tabulka má jeden primární klíč.

Reg. číslo	Licence	Oddíl
8401	C	Nová Paka
8151	D	Hostinné
9001	A	Nová Paka

Tab. 4: Dekompozitovaná tabulka – reg. číslo.

Oddíl	Zkratka oddílu
Nová Paka	NPA
Hostinné	HOS

Tab. 5: Dekompozitovaná tabulka – oddíl.

7.4 Třetí normální forma (3. NF)

Tabulka je ve třetí normální formě tehdy, pokud:

- Je ve druhé normální formě.
- Neexistují závislosti neklíčových sloupců tabulky.

Jako příklad použiji tabulku oddílu, uvedenou jako špatně navrženou pro druhou normální formu (Tab. 3: Špatně navržená tabulka 2. NF). Výše uvedená definice se zde týká sloupců Oddíl a Registrační číslo. Seznam oddílů a seznam členů jsou dva odlišné informační zdroje. Proto vytvoříme tabulku oddílů, na kterou se pak budeme odkazovat cizím klíčem (odkazem).

7.5 Čtvrtá normální forma (4. NF)

Tabulka je ve čtvrté normální formě tehdy, pokud je ve třetí normální formě a popisuje jen jeden jediný fakt nebo souvislost.

7.6 Pátá normální forma (5. NF)

Tabulka je v páté normální formě tehdy, pokud je ve čtvrté normální formě a není možné do ní přidat nový sloupec či skupinu sloupců bez toho, aniž by se tato tabulka rozpadla na několik dílčích tabulek.

Závěrem kapitoly bych chtěl upozornit, že ne vždy je vhodné snažit se dosáhnout co nejvyšší normální formy. Vedle správnosti navržení je důležité sledovat i rychlost (složitost) aplikace. Je zbytečné udržovat a hlavně procházet všechny PSČ v České republice nebo pamatovat si všechny svátky ke každému dni, pokud má firma 20 zaměstnanců. Pro normální použití v běžných aplikacích stačí 3. NF (většinou i 2. NF).

8. Projekt přihlašovacího systému

V této kapitole bych chtěl ukázat důvod vzniku přihlašovacího systému. Vedla k tomu potřeba zefektivnit podávání přihlášek na jednotlivé závody. Zlepšení by se mělo projevit jak v časové, tak i finanční úspoře.

8.1. Orientační běh

Orientační běh je sport, při kterém závodník musí v co nejkratším čase sesbírat kontroly zakreslené v mapě. Cesta mezi kontrolami je libovolná a závisí čistě na rozhodnutí závodníka, kudy se k další kontrole dostane. Důležité je však sesbírat všechny kontroly v daném pořadí. Na závodě je vypsáno několik kategorií. Základní dělení je na mužské a ženské kategorie. Dále se obě kategorie dělí dle věku a každá věková kategorie se dělí ještě podle výkonnosti. Na závodě je tak vypsáno okolo 40 kategorií.

8.2. Důvod vzniku aplikace a požadavky

Tento systém vznikl, protože dosavadní přihlašování na závody bylo nepohodlné, málo flexibilní a v neposlední řadě i zbytečně finančně nákladné. Každý člen si musel hlídat, na které závody z kalendáře ČSOB chce jet a do které kategorie se chce přihlásit. Před každým závodem musel volat vedoucímu oddílu, který si každého člena zapsal a následně sestavil přihlášku, kterou poslal emailem pořadateli závodu. Nezřídka se stávalo, že se člen zapomněl přihlásit, a musel platit zvýšené startovné, nebo na daný závod nemohl vůbec nastoupit. Když si někdo rozmyslel účast na závodě, musel opětovně volat vedoucímu, který si ho v přihlášce vyškrtl. Další nevýhodou bylo volání ostatním závodníkům kvůli dohodnutí kategorie, která se poběží.

Protože se oddíl rozrůstá a toto řešení nebylo již únosné, po vzoru velkých oddílů jsme se dohodli o zavedení podobného systému, jako používají ony. Následné hledání vhodného, již existujícího řešení nebylo úspěšné. Systémy byly buď málo propracované, nebo naopak až příliš složité. Našimi hlavními zásadami jsou

funkčnost, jednoduchost, přehlednost a velmi důležitá samostatnost programu. Tou je myšleno, aby během roku bylo zapotřebí co nejméně administrátorských zásahů (co nejméně času stráveného s jeho údržbou a aktualizací). Zde nevyhovovala většina propracovaných existujících programů. Proto jsem začal tvořit systém od úplných začátků přesně podle našich požadavků.

9. Vývoj aplikace přihlašovacího systému

Od začátku vývoje jsem se snažil o co největší použitelnost a záleželo mně na tom, aby systém byl plně funkční a vyhovoval všem, kteří ho budou používat. Proto jsem naprogramoval hrubý funkční základ a umístil ho na web. Systém běžel zároveň s reálnými přihláškami. Každý mohl vyjádřit svoji připomínku nebo nápad, kterým by se aplikace vylepšila. Připomínek se sešlo opravdu hodně, na některé se ani v časové tísni nedostalo, a už teď je jisté, že program se bude dále zdokonalovat a zlepšovat.

9.1 Návrh databáze

Databáze přihlašovacího systému běží na MySQL ve verzi 5.0.32. Zde popisuji jednotlivé tabulky. Diagram databázové struktury je v příloze A.

Popis jednotlivých tabulek:

Clenove – tabulka obsahuje všechny údaje týkající se uživatelů včetně jejich nastavení účtu. Obsahuje údaje o všech třech typech účtu (normální uživatel, administrátor, účet pro odeslání přihlášek).

Zavody – tabulka obsahuje údaje o závodech. Závody, které už proběhly, se nemažou a zůstávají uloženy. Systém je ovšem nezobrazuje.

Poradatele – zde jsou uloženy záznamy o všech oddílech, které pořádají závody. Tabulka obsahuje jméno oddílu, zkratku oddílu a odkaz na oddíl. Odkaz je použit všude, kde je oddíl uveden. Uživatel tak má ihned přístup ke stránkám závodu, např. pro zjištění kategorie.

Poznamky – každý uživatel může zadat poznámku k přihlášce. Ta je uložena v tabulce. Ta přiřazuje jednotlivé poznámky uživatelů k daným závodům. Hlavní poznámka je typu text, takže není délkou nikterak omezena.

Prihlaseni – tato tabulka zaznamenává jednotlivá přihlášení. Obsahuje údaje o tom, kdo se přihlásil na konkrétní závod a kterou kategorii si vybral.

9.2. Návrh a základní popis programu

K psaní programů jsem používal dva editory. Pod Windows jsem používal vývojový editor PSPad a pod Linuxem editor Quanta. Oba dva podporují psaní přímo na FTP serveru. Vývoj na FTP serveru má hned několik výhod. Při své práci jsem nemusel přenášet zdrojový kód mezi různými počítači a měl jsem k dispozici vždy nejnovější verzi. Největší výhodou ale je, že kód se píše přímo na serveru, kde jednotlivé skripty poběží. Zamezí se tím problémům mezi nekompatibilními verzemi jednotlivých programů (hlavně nastavení PHP).

Aplikace obsahuje v kořenovém adresáři dva soubory a dva adresáře. První soubor je index.php. Ten obsahuje úvodní stranu celé aplikace. Druhým souborem je skript.php. Tento skript spouští Cron každý den a rozesílá jednotlivým členům upozornění na přihlášky. Dále je zde adresář obrazky, který obsahuje všechny obrázky v aplikaci. Druhý adresář slozky obsahuje všechny zbylé kódy aplikace. Přihlašovací systém běží na adrese: bakalarka.npa.cz.

9.3. Úvodní strana

Na úvodní stránce se závodník dozví, kdo z ostatních členů jede na který závod. Je zde přehled všech závodů a k nim vypsané přihlášky jednotlivých závodníků. Je to vstupní místo pro přihlášení do aplikace.

9.3.1. Popis úvodní strany

V levém bočním pruhu je přehled nejbližších šesti přihlášek. Tato informace je velmi důležitá, proto je zde uvedena. Celý systém je postaven na včasném upozornění závodníka, aby se stihl rozhodnout, kterou kategorii poběží, a včasném přihlášení na závod. Výpis šesti závodů je zde proto, že se může vyskytnout situace, kdy více závodů má termín přihlášky v jeden den. Proto je důležité, aby byl přehled znám s časovou rezervou, kdy končí jednotlivé termíny přihlášek.

V hlavním těle úvodní stránky je přehled všech závodů. Každý závod lze rozbalit, abychom viděli závodníky, kteří se na daný závod zapsali. To se provádí znakem [+], který je u každého závodu. Standardně se při prvním najetí rozbalí tři závody. Z hlediska přehlednosti je to optimální nastavení, protože závody bývají většinou po týdnu, a časová rezerva je tedy dostatečná. Přesto lze dorozbalit kterýkoliv závod. Pro rychlé rozbalení všech závodů je zde vlevo nahoře tlačítko rozbal vše (zabal vše).

Každý závod již mohl být uzavřen. To znamená situaci, kdy přihláška na závod byla již odeslána, a nelze se již přihlásit ani odhlásit. To je značeno podbarvením celého závodu světle červenou barvou. U neodeslaných závodů je podbarvení světle zelené.

Obr. 3: Ukázka aplikace – hlavní strana.

9.3.2. Programové řešení hlavní strany

Hlavní strukturou jsou dva vnořené cykly, z nichž jeden načítá budoucí závody a druhý pro konkrétní závod vypíše přihlášené závodníky. To, který závod se rozbalí, je dáno proměnou typu pole, ve kterém jsou uloženy ID rozbalených závodů. Pole je přenášeno pomocí relací. Přidávané a ubírané položky jsou přenášeny pomocí metody GET. Když metoda nese nějaké číslo, zkontroluje se, zda je v poli. Pokud ne, zapíše se na konec pole. V opačném případě je položka v poli smazána (závod se zabalí). Ukázka kódu:

```
$zmena= false;  //nastavení hodnoty indikující změnu
$cislo= $_GET["cislo"];    // předá se číslo
$pole= $_SESSION['pole'];  //předá se pole z relace
for($i=0; $i < count($pole); $i++)    //pro celé pole
{
    if($pole[$i] == $cislo)     //když najde - UBÍRA SE
    {
        array_splice($pole, $i, 1);
        $zmena= true;      //zaznamená se změna
    }
}
if($zmena == false)  //když nenajde - PŘIDÁVÁ SE
{  $pole[]= $cislo;  }
```

9.4. Uživatelské rozhraní

Uživatelský účet slouží všem registrovaným závodníkům pro přihlášení na jednotlivé závody. Je zde uveden přehled přihlášených závodů a možnost nastavení účtu daného závodníka

9.4.1. Popis uživatelského rozhraní

Každý uživatel se přihlašuje pod svým registračním číslem a heslem. První stránka, na kterou se dostane, vypisuje přihlášené závody. Zde vidí výpis přihlášených závodů a zvolenou kategorii. Kromě nejdůležitější kategorie je zde přehled o všech údajích týkajících se závodu (shromaždiště, jméno závodu,

pořádající oddíl). Oddíl je vypsán ve zkratce, ale po najetí myší na odkaz se zobrazí celý název. Kromě toho po kliku na oddíl se v novém okně otevře stránka pořádajícího oddílu. To samé platí pro název závodu, kde se uživatel dostane na stránku závodu. Zde si může vybrat jednu z vypsaných kategorií. Jsou zde uvedeny i dvě poznámky. První se týká poznámky k závodu. Zde se mohou uvádět informace typu jak se pojede na závod nebo kde se bude spát. Prostě to, co je potřeba oznámit členům, kteří si závod zapisují. Druhou poznámku si může zadat uživatel, potřebuje-li něco sdělit tomu, kdo daný závod bude přihlašovat. Příkladem může být situace, kdy chce někdo přihlásit ke konci startovního pole nebo kdy chce na vícedenních závodech přihlásit jen některé etapy. Obě poznámky jsou nepovinné a nemusí se zadávat. Po kliku na každou z nich se daná poznámka otevře v novém, samostatném okně.

Pokud byl závod již odeslán, zmizí checkbox před daným závodem a ten již nelze odhlásit ani měnit jiné údaje. Pod seznamem závodů se nachází oblast, kde lze označeným závodům měnit přihlášenou kategorii, smazat poznámku nebo závody odhlásit.

Další položkou v menu uživatele je položka „přihlásit závod“. Ta vypadá podobně jako předchozí stránka, ale jsou zde uvedeny závody, na které se lze přihlásit (uživatel ještě není přihlášen a není odeslána přihláška). Zde se vedla mezi závodníky největší debata, jak má být tento seznam závodů řazen. První možnost je podle data konání závodu, druhá dle data přihlášky daného závodu. Obě varianty mají své opodstatnění. Počet zastánců jednotlivých skupin se rozdělil zhruba na polovinu. Proto jsem střet názorů vyřešil tak, že se do databáze uloží poslední nastavení, které uživatel provedl. Při příštím přihlášení má uživatel seřazeny závody tak, jak mu to vyhovuje. Do databáze se ovšem ukládá jenom dle čeho se má řadit. To znamená, že se řadí buď podle data závodu nebo data přihlášek, ale vždy vzestupně. Sestupné pořadí zde nemá žádný význam.

Obr. 4: Ukázka aplikace – přihlášené závody.

9.4.2. Implementace uživatelského rozhraní

Zabezpečení přístupu - zabezpečení jednotlivých účtů je řešeno pomocí relací. Po přihlášení se v souboru login.php přiřadí číslo, kterým jsou identifikovány jednotlivé typy účtu (uživatelé mají jedničku). Na začátku každého souboru je pak toto číslo kontrolováno a v případě nepovoleného vstupu je návštěvník přesměrován na přihlašovací obrazovku.

```
session_start();
if($_SESSION["login"] != 1)
{     echo 'Nejste přihlášen:
            <a href="login.php">přihlásit</a>';
      Exit;
}
```

Řazení závodů - výše jsem se zmiňoval o tom, že si aplikace pamatuje, jak má uživatel seřazeny závody. Tedy buď podle data konání závodu, nebo dle data přihlášky. Je to provedeno tak, že se hlídá parametr „dle“ v metodě GET. Když

obsahuje určitá data, uživatel provede řazení a tato data se zapíší do databáze a nastaví se, dle jakého kriteria se výpis seřadí. Když je parametr „dle“ prázdný, uživatel přistupuje na stránku poprvé a řazení se načte z databáze. Ukázka kódu:

```
if(isset($_GET["dle"]))  {      //je provedeno řazení
     $dle=$_GET["dle"];
     if($dle == "datum")        //zjišťuje se nastavení
     $pom= "z";      //řazení - závod
     else
     $pom= "p";      //řazení - přihláška
mysql_query("UPDATE clenove SET razeni = '$pom'
            WHERE reg_cislo= '$uzivatel'");  }
else  {    //jestliže není řazení - načítá z databáze
$sql= mysql_query("SELECT clenove.razeni FROM clenove
                  WHERE  reg_cislo= '$uzivatel'");
   $vysledek= MySQL_Fetch_Array($sql);
   $dle= $vysledek["razeni"];
   if ($dle == 'z')
$dle= "datum";       //razeni dle data
   else
$dle= "dat_prihl";   //razeni dle prihlasky
}
```

Ukládání hesla - na stránce nastavení si uživatel může měnit heslo pro přihlášení. Heslo se ukládá do databáze jako otisk pomocí algoritmu SHA1. Tím pádem heslo nikdo nevidí (administrátorský uživatel ani administrátor spravující databázi). Má to ovšem i své nevýhody. Když uživatel zapomene své heslo, nelze ho již zpětně zjistit a musí se vygenerovat heslo nové. Ukázka kódu ukládající heslo:

```
                //vytvoří se otisk
$heslo= sha1(mysql_escape_string ($_POST['nove']));
                //provede se update v databázi
$vysledek= mysql_query("UPDATE clenove SET heslo='$heslo'
                        WHERE reg_cislo='$uzivatel'");
$hlaska[]= 'Heslo bylo úspěšně změněno !';
```

Email při vstupu – uživatel si může nastavit, jestli chce poslat email při přihlášení do systému. Tato funkce je zde jako kontrolní a slouží pro bezpečnost účtu. Lepší bezpečnost účtu zajišťuje i funkce, která odešle email při určitém počtu špatně zadaných pokusů o přihlášení. Při každém špatně zadaném pokusu se v databázi odečte jeden pokus pro daného uživatele. Každý uživatel si tuto funkci může aktivovat (deaktivovat) a zadat, při kolika pokusech se má upozornění odeslat. Při dosažení tohoto čísla se vygeneruje email, který je zaslán uživateli, a hodnota je inicializována na hodnotu definovanou uživatelem. Příklad tohoto kódu:

```
if(($zbyva==0) and ($kdy_email[1]=='a')) //má se poslat email
{
//vygenerování proměnných v emailu
$datum = Date("j/m/Y - H:i:s", Time());
$text= "
Datum: $datum
 Uzivatel: NPA $user
 Počet neúspěšně zadaných hesel ($kolik_pokusu) byl vyčerpán.
“;
 //volána funkce email
 mail($email, "Špatná hesla k NPA.CZ", $text);
 mysql_query("UPDATE clenove SET pokusu_zbyva= '$kolik_pokusu'
         WHERE reg_cislo= '$user'");
}
else { //odečítá jeden neplatný pokus
   $zbyva--;    //počet zbývajících pokusů je snížen
   mysql_query("UPDATE clenove SET pokusu_zbyva= '$zbyva'
                WHERE reg_cislo= '$user'");
}
```

9.5 Administrátorský účet

Uživatelé s tímto oprávněním zadávají do systému všechna data. Jedná se o pořadatelské oddíly, uživatelské účty a závody. Tento účet by měl být využíván co nejméně, zejména, když je nový rozpis závodů nebo přibude další člen do oddílu.

9.5.1. Popis administrátorského rozhraní

Po přihlášení se zobrazí stránka s přehledem závodů. Zde je přehled všech údajů o každém závodě. Odkaz závodu je vzhledem k malému množství místa upraven tak, že je ze začátku odříznut údaj obsahující http://www. Pořadí lze měnit dle data závodu, jména závodu a data přihlášky. Každé datum v celém systému je opatřeno číslem, které udává počet dní zbývajících do daného data. Číslo je uvedeno v závorkách. To umožňuje rychlejší orientaci v čase.

Pod výpisem údajů o závodech je formulář, kde se dají měnit všechny údaje vypsaných závodů. Vybrané závody se označí pomocí checkboxu. Vše, co se dá změnit, je ošetřeno tak, aby se do databáze nedostala žádná nekorektní data, tj. hlídání typů proměnných, použití regulárních výrazů, hlídání hodnoty proměnné.

Další v pořadí menu je položka přehled uživatelů. Ta obsahuje přehled všech členů se všemi údaji o nich. Jedinou položkou, která zde není uvedena, je heslo. To je totiž uvedeno jako otisk a ten k ničemu nepotřebujeme. Dole ve formuláři ovšem heslo měnit jde, to pro případ, že by některý uživatel heslo zapomněl. Jinak je celá stránka řešena podobně jako přehled závodů.

Zde vysvětlím, jak jsou řešena přístupová práva uživatelů. Pro každý účet je zadáno, o jaký se jedná typ (v databázi označeno písmenkem: U – uživatel, R – administrator, O – pro odeslání přihlášek). Při vytváření lze zadat kterýkoliv typ účtu, tudíž je možné, aby existovalo více účtů, které spravují databázi nebo které odesílají přihlášky. Proto databázi a přihlášky může obsluhovat více lidí.

Přidávání závodů, členů a oddílů je řešeno v následujících třech položkách menu. Zde jsou formuláře, kde se zadávají povinné i nepovinné údaje. Všechny položky jsou ošetřeny proti zadání špatných údajů a při každé akci provedené uživatelem se nad formulářem zobrazí výčet provedených či chybných hlášení. Každá položka obsahuje nápovědu. Ta se zobrazí při najetí myší nad příslušné zadávací políčko. Takto jsou ošetřena všechna vstupní políčka včetně tlačítek v celé aplikaci.

Obr. 5: Ukázka aplikace – administrátorský účet.

9.5.2. Implementace administrátorského účtu

I když pouze administrátorský účet vkládá časové údaje, ošetřil jsem je regulárním výrazem. Data se vkládají ve formátu (rrrr-mm-dd). To je ošetřeno regulárním výrazem:

```
if(EregI("^[0-9]{4}-[0-9]{2}-[0-9]{2}$", $datum))
```

Chtěl jsem, aby každá akce provedená uživatelem vyvolala příslušnou hlášku o provedení činnosti nebo nastalé chybě. Protože uživatel může editovat více údajů, musí se vypsat i více informativních hlášení. Hlášky tedy ukládám do pole. Příkladem takové akce může být změna údajů u vybraných položek nebo přidání nových položek. Pole, ve kterém jsou uloženy hlášky, je zkontrolováno a v případě, že není prázdné, se hlášky vypíšou.

```
if(count($hlaska) != 0) {       //jestliže pole není prázdné
echo'<div style="_text-align: center;">// formátovací zančky
<div class="obal_akce">';
for ($i=0; $i<count($hlaska); $i++) { //pro celé pole
 echo'<div class="akce" style=" text-size: 8px;">
                - '.$hlaska[$i].'</div>';
}
echo '</div></div>'; }          //konec formátovacích značek
```

9.6. Účet odesílatele

Tento účet je používán k odesílání přihlášek na závody. Uživatel odesílá přihlášku pořadateli závodu, čímž se závod automaticky uzavře. Může si přečíst poznámky k přihlášce od uživatelů a vhodně na ně reagovat.

9.6.1 Popis rozhraní pro odesílatelský účet

Při přihlášení se zobrazí přehled závodů. Zde jsou uvedeny všechny údaje o závodě včetně poznámky závodu a poznámky od uživatele. Poznámka závodu je zde uvedena ve zkrácené formě 10 znaků z důvodu nedostatku místa. Při najetí myší se ale v kontextovém menu zobrazí poznámka celá nebo po kliknutí na tuto poznámku se otevře v novém okně. Poznámka od uživatelů je zde zobrazena ve zkrácené formě [+]. Rovněž funguje jako odkaz, kde se v novém okně zobrazí všechny poznámky od uživatelů, které souvisejí se závodem.

Fakt, že byl závod již odeslán, indikují dvě věci. Jednak je to odkaz, který je u odeslaných závodů „Podívej“. U neodeslaných závodů je to pak „Odešli“. Druhým znakem odeslaných závodů je checkbox, který je umístěn na začátku řádku. Ten se může využít pro případ, že by se závod znovu otevíral pro přihlášky. Může se stát, že pořadatel posune termín přihlášek. V tom případě uživatel, který znovu zpřístupňuje závod pro přihlášky, zatrhne příslušný checkbox a stiskne tlačítko Zpřístupni. Může změnit i poznámku závodu (př.: termín je prodloužen). Pro samotné odeslání závodů je zde odkaz „Odešli“.

Na této stránce je úplně nahoře uveden závod, o který se jedná. Hned pod tím je uveden výčet členů, kteří jsou na daný závod přihlášeni i s jejich vybranou kategorií a poznámkou. Pod tímto seznamem uživatelů je formulář, sloužící pro odeslání závodů. Existují dvě možnosti, jak závod odeslat - buď stáhnout přihlášku jako soubor, ten se otevře v novém okně a závod se tím automaticky uzamkne, nebo rovnou odeslat email pořadateli závodu. Soubor s přihláškou se pošle jako příloha tohoto emailu, přesně tak, jak je to dáno normou ČSOB. Na formuláři se nachází políčko email příjemce. Na tuto adresu dojde přihláška. Dále zde máme políčka pro zadání předmětu emailu a textu emailu. Při odeslání emailu je rovněž závod uzavřen pro další přihlášky.

Obr. 6: Ukázka aplikace – odeslání závodu.

9.6.2. Programové řešení odesílatelského účtu

Pro odesílání přihlášky jsem použil třídu mime_mail.inc. Tuto třídu jsem našel na internetovém serveru interval.cz [15], který ji převzal z nakladatelství Wrox Press. Při každém otevření jednotlivých závodů se vytvoří soubor data.txt. Ten obsahuje danou přihlášku a je načten do přílohy. Ukázka kódu, nastavujícího třídu mime_mail.inc pro odesílání emailů:

```
$mail=new mime_mail;  //vytvoření nové instance
//nastaví, od koho se má email poslat
$mail->from = $_POST["email_od"];
//adresa,na kterou se má email poslat
$mail->to = $_POST["email"];
$mail->subject = $_POST["predmet_emailu"]; //předmět emailu
$mail->body = $_POST["text_emailu"];  //tělo textu
$mail->add_attachment($data, "data.txt", "data.txt");
     //načtení přílohy


$mail->send();  //poslání emailu
```

Kontrola emailu – Pro všechny uživatele jde nastavit odeslání emailu při určitém počtu dnů před koncem jednotlivých přihlášek. To zajišťuje skript, který je každý den spouštěn Cronem na serveru www.c4.cz. Ten je tvořen jedním cyklem, který vybírá postupně všechny závodníky, kteří mají aktivní upozorňování, a zjistí, kolik je nastaveno dnů. Poté projde všechny závody a případně odešle email. Email jde samozřejmě měnit v nastavení. Správnost emailové adresy je kontrolována regulárním výrazem.Ukázka regulárního výrazu:

```
if(EregI("^[a-z0-9]+[a-z0-9\._-]*[a-z0-9]+@[a-z0-9]+[a-z0-
9\._-]*[a-z0-9]+\.[a-z]{2,3}$", $email) && !EregI("\.{2,}",
$email) && !EregI("_{2,}", $email) && !EregI("-{2,}", $email))
```

Tento výraz říká, že na začátku adresy musí být jakékoliv písmeno nebo číslice. Následovat mohou slova oddělená speciálním znakem a před zavináčem je opět pouze písmeno nebo číslice. Dále je zavináč následovaný výrazem ze začátku. Konec je tečka a za ní doména tvořená dvěma nebo třemi písmenky. Konec regulárního výrazu tvoří skutečnost, že se v adrese nesmí vyskytovat dva speciální znaky za sebou. Tento výraz vrací true v případě správnosti, jinak vrací false.

10. Závěr

Cílem této práce bylo vytvoření přihlašovacího systému pro oddíl orientačního běhu. Práce popisuje celý vývoj přihlašovacího systému od požadavků přes návrh až po samotnou implementaci.

V úvodu své bakalářské práce popisuji použité technologie. Byly zde popsány a porovnány nejběžnější programovací jazyky a databázové systémy. Popsal jsem normální formy, podle kterých by se měl řídit návrh databází u podobných systémů. Rovněž zde byly uvedeny technologie, které využívám ve svém programu. Jedná se o hashovací funkce SHA1 porovnanou s funkcí MD5. Dále byly uvedeny cookies, relace, regulární výrazy a SQL injection.

Návrh celé aplikace byl vytvářen podle požadavků a návrhů členů orientačního klubu SK LOB Nová Paka. Umožňuje jednotlivým členům zadávání přihlášek na vybraný závod a snaží se o co největší „komfort“ při přihlašování. Grafický design celé aplikace je složen z barev,které má oddíl jako klubové. Systém splňuje požadavky, které na něj byly kladeny. Nicméně potenciál využití tohoto programu je daleko vyšší. Byly vzneseny další návrhy jako sledování statistik a výsledků jednotlivých členů po závodě. Nejedná se tedy o konečné řešení a systém se bude dále vyvíjet. Při dalším vývoji by se ale měl brát ohled na to, že dosud byla aplikace vyvíjena pro minimální úsilí potřebné k jejímu provozu. Závody se zadávají dvakrát do roka a rovněž uživatelé tráví minimum času při přihlášení. Problémem, který by mohl nastat při dalším vývoji, je formát data. Při ukládání data závodu a data přihlášky je použit formát (rrrr-mm-dd). Pro dosavadní účely tento formát zcela postačuje (řazení, získání jednotlivých dnů, měsíců), ale lepší by byl formát UNIXového časového razítka. UNIXové razítko poskytuje rychlejší práci s časem a nabízí rovněž větší možnosti.

Seznam použité literatury

[1] KOFLER, M. BERND O. *PHP 5 a MySQL 5 Průvodce webového programátora*. 1. vydání. Computer Press, a.s., 2007. 608 s. ISBN 978-80-251-1813-9

[2] LACKO, Luboslav. *Oracle Správa, programování a použití databázového systému.* 1. vydání. Computer Cpress a.s., 2003. 466 s. ISBN 80-7226-699-3

[3] *Webový server - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-03-31]. <http://cs.wikipedia.org/wiki/Webov%C3%BD_server>.

[4] *Webhosting C4* [online]. 2006. <http://www.webhosting-c4.cz/>.

[5] *HyperText Markup Language - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-03-05]. <http://cs.wikipedia.org/wiki/HyperText_Markup_Language>.

[6] *Hypertext Transfer Protocol - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-05-11]. <http://cs.wikipedia.org/wiki/HTTP>.

[7] DUŠAN, Janovský. *HTTP protokol* [online]. 2008 [cit. 200-04-26]. <http://www.jakpsatweb.cz/server/http-protokol.html>.

[8] KOSINA, Pavel. *Python - popis jazyka - python, úvod do jazyka, Guido van Rossum* [online]. 2005 [cit. 2005-05-09]. <http://programujte.com/index.php?akce=clanek&cl=1970010106-python-popis-jazyka>.

[9] DAŘENA, František. *Perl* [online]. 2003 <https://akela.mendelu.cz/~darena/Perl/>.

[10] *PHP - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-04-26]. <http://cs.wikipedia.org/wiki/PHP>.

[11] *MySQL - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-04-26]. <http://cs.wikipedia.org/wiki/MySQL>.

[12] HABRMAN, Robert. *PostgreSQL, Úvod* [online]. 2007 [cit. 2007-07-11]. <http://www.owebu.cz/databaze/vypis.php?clanek=1634>.

[13] BAAR, Ondřej. *Nebezpečnost MD5 - chraňte svá hesla!* [online]. 2007 [cit. 2007-06-27]. <http://www.owebu.cz/bezpecnost/vypis.php?clanek=1156>.

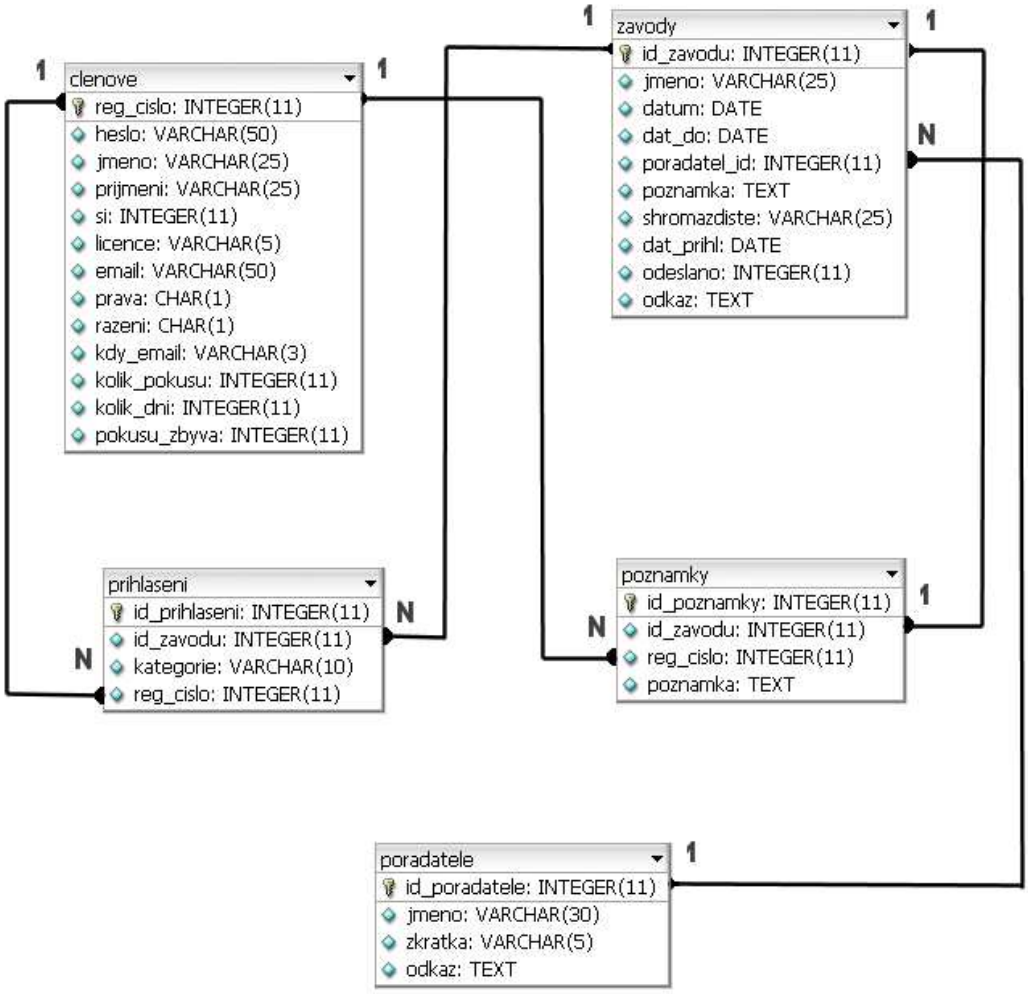
[14] *Secure Hash Algorithm - Wikipedie, otevřená encyklopedie* [online]. 2008 [cit. 2008-04-03]. <http://cs.wikipedia.org/wiki/Secure_Hash_Algorithm>.

[15] KAŠÍK, Martin. *Jak v PHP poslat e-mail s přílohou pomocí MIME třídy -- PHP -- Vývoj aplikací -- Interval.cz* [online]. 2002 [cit. 2002-02-11]. <http://interval.cz/clanky/jak-v-php-poslat-e-mail-s-prilohou-pomoci-mime-tridy/>.

[16] WELTON, David. *Programming Language Popularity* [online]. 2004 [cit. 2005-10-20]. <http://www.welton.it/articles/language_popularity.html >

Příloha A

Relace mezi databázovými tabulkami



Příloha B

Ukázka přihlášky na závody orientačního běhu dle normy ČSOB.

Reg.č.	Kat.	SI čip	Příjmení,jméno	Lic.
NPA9101	D18A	0000009707	Šmidrkalová Monika	A
NPA8151	D21A	0000009701	Píšová Monika	C
NPA7551	D21C	0000009717	Pavelcová Gabriela	C
NPA7001	D45A	0000009712	Hušková Eva	A
NPA5502	D45C	0000009714	Nováková Jana	B
NPA5501	D45D	0000009713	Kalibánová Jiřina	C
NPA9501	H14C	0000009706	Kosina Jakub	C
NPA9001	H18B	0000009705	Horák Jan	C
NPA9003	H18C	0000009711	Stuchlík Jaroslav	C
NPA8402	H21A	0000009704	Prášil Ondřej	C
NPA7701	H21B	0000009699	Novák Zdeněk	D
NPA8401	H21C	0000009700	Bejr Zdeněk	A
NPA9002	H21C	0000009710	Brendl Radek	C
NPA7601	H21D	0000009718	Pavelec Tomáš	D
NPA7502	H21E	0000009703	Novák Jiří	R
NPA5101	H45C	0000009716	Kalibán Jaroslav	B
NPA5001	H50C	0000009708	Novák Karel	C
NPA4901	H55C	0000009715	Prášil Jan	C